CPSC 3750
Assignment 1
Luke Leontowich
001217324

1)

- Playing Soccer:
  - The expected utility: Number of goals scored
  - The environment: Soccer field, players, soccer ball
  - The actions: Kicking, running, passing, shooting, planning plays to score
  - The sensors: Cameras, tracking devices for players and ball
  - Fully Observable: All players, and ball are visible
  - Multi-agent: Multiple players and a soccer ball
  - Stochastic: Injuries, different player skills and strength. Makes it hard to predict the game. Goalies will a save percentage that could be based on location of shot. For example, goalie A saves 10% of shots that are in the top right corner.
  - Semi-dynamic: The field stays the same, but the players are constantly moving. A player's performance will be different throughout the game based on stamina and adrenaline.
  - Continuous: the state of the game is continuously changing and has infinitely many states.
- Exploring the subsurface of the an ocean:
  - The expected utility: correct analysis of ocean foliage, animals, sea floor terrain, water quality, etc.
  - The environment: The ocean foliage, animals, sea floor terrain, water quality, etc.
  - The actions: Scanning, moving about the ocean.
  - The sensors: Camera's, radar, sonar, other scientific devices to measure desired ocean factor.
  - Partially: Possible, though improbable to measure all the aspects of the ocean .
  - Multi-agent: Many different attributes to the ocean.
  - Stochastic: Too complex to observe all unobserved attributes.
  - Dynamic: Environment is always changing.
  - Continuous: The state of the environment is always changing.
- Bidding on an item at an auction:
  - The expected utlility: Getting the item for the cheapest amount possible
  - The environment: Yourself, other bidders, the item
  - The actions: bidding, waiting
  - The sensors: Timer, people's bid amount
  - Partially observable: You don't know how much each person is willing to pay
  - Multi-agent: Could be many different people bidding on the item
  - Deterministic: The state of the bid is determined by the previous state. If the the bid increased, then the next state will be bid increased or item sold
  - Static: The states are the current bid or sold
  - Discrete: Their are a finite number of states and a discrete set of percepts and action

CPSC 3750
Assignment 1
Luke Leontowich
001217324

2)

a) Using only four colours, you have to colour a planar map in such a way that no two adjacent regions have the same colour:

- States: the state is the colours that have been coloured in so far
- Initial State: Nothing has been coloured yet
- Action: Check adjacent regions of what their colours are, and then try each colour that will work. A transition will work if the colour chosen is not the same as its adjacent colours
- Transition: Colour in a region
- Goal States: The planar map is completely coloured in such that no adjacent regions are the same colour
- Action Cost: The cost is the cost of checking all the adjacent regions of the region to be coloured

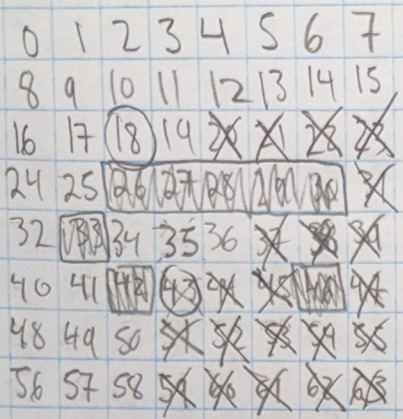
b) You have a program that outputs the message “illegal input record” when fed a certain file of input records. You know that the processing of each record is independent of the other records. You want to discover what record is illegal.

- States: Legal or illegal
- Initial State: Legal
- Action: change state to illegal or stay in state legal
- Transition: If illegal return “illegal input record” else keep going until end of input files
- Goal States: Illegal input state or empty set of files to process
- Action Cost: Cost O(how every long it takes to check a file)

c) You have three jugs measuring 12 gallons, 8 gallons and 3 gallons, and a water faucet. You can fill the jug up, empty it out from one to another or onto the ground. You need to measure out exactly one gallon.

- State: The measurement of total water at a given instance
- Initial State: 0 gallons
- Actions: Fill 1 of the jugs, empty a jug into another, empty jug onto ground
- Transition: One of the jugs going from empty to full, One jug getting filled by another, one jug getting dumped on the ground
- Goal States: 1 Gallon of water total
- Action Costs: Each action costs the amount of water exhanged

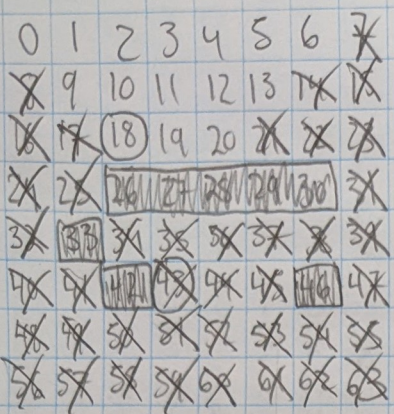
3,a)



Node	frontier (stack) (stack.top on right side)
[]	[43]
43	[35, 44, 51]
51	[35, 44, 50, 52, 59]
59	[35, 44, 50, 52, 58, 60]
60	[35, 44, 50, 52, 58, 60, 52, 61]
61	[35, 44, 50, 52, 58, 60, 52, 53, 62]
62	[35, 44, 50, 52, 58, 60, 52, 53, 54, 63]
63	[35, 44, 50, 52, 58, 60, 52, 53, 54, 55]
55	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 54]
54	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 53]
53	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 52]
52	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 44]
44	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 45]
45	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 37]
37	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 36, 38]
38	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 36, 39]
39	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 36, 31, 47]
47	["]
31	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 36, 23]
23	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 36, 15, 22]
22	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 36, 15, 14, 21]
21	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 36, 15, 14, 13, 20]
20	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 36, 15, 14, 13, 12, 19]
19	[35, 44, 50, 52, 58, 60, 52, 53, 54, 47, 45, 36, 36, 15, 14, 13, 12, 11, 18]
18	

found the goal. The order of nodes is
43-51-59-60-61-62-63-55-54-53-52-44-45-37-38-39-47-31-23-22-21-20-19-18

3.b)



Node	frontier (queue) (queue.front on left)
[7	[43]
43	[35, 44, 51]
35	[44, 51, 34, 36]
44	[51, 34, 36, 36, 45, 52]
51	[34, 36, 36, 45, 52, 50, 52, 59]
34	[36, 36, 45, 52, 50, 52, 59]
36	[45, 52, 50, 52, 59, 37]
45	[52, 50, 52, 59, 37, 37, 53]
52	[50, 52, 59, 37, 37, 53, 60]
50	[52, 59, 37, 37, 53, 60, 49, 58]
59	[37, 53, 60, 49, 58, 58, 60]
37	[53, 60, 49, 58, 60, 38]
53	[60, 49, 58, 60, 38, 54, 61]
60	[49, 58, 60, 38, 54, 61, 61]
49	[58, 60, 38, 54, 61, 41, 48, 57]
58	[60, 38, 54, 61, 41, 48, 57, 57]
38	[54, 61, 41, 48, 57, 39]
54	[61, 41, 48, 57, 39, 55, 62]
61	[41, 48, 57, 39, 53, 62, 62]
41	[48, 57, 39, 53, 62, 40]
48	[57, 39, 53, 62, 40, 56]
57	[39, 53, 62, 40, 56, 56]
39	[53, 62, 40, 56, 31, 47]
62	[40, 56, 31, 47, 63]
40	[56, 31, 47, 63, 32]
56	[31, 47, 63, 32]
31	[47, 63, 32, 23]
47	[63, 32, 23, 55]

Node	Frontier
63	[32, 23, 55, 55]
32	[23, 55, 24]
23	[55, 24, 15, 22]
55	[24, 15, 22]
24	[15, 22, 16, 25]
15	[22, 16, 25, 7, 14]
22	[16, 25, 7, 14, 14, 21]
16	[25, 7, 14, 21, 8, 17]
25	[7, 14, 21, 8, 17, 17]
7	[14, 21, 8, 17, 6]
14	[21, 8, 17, 6, 6, 13]
21	[8, 17, 6, 13, 13, 20]
8	[17, 6, 13, 20, 0, 9]
17	[6, 13, 20, 0, 9, 9, 18]

18 is goal so we are done
order is the order of nodes in nodes column.

CPSC 3750
Assignment 1
Luke Leontowich
001217324

3) c) For C, there where way too many nodes to keep track of so I wrote a program and will post the results of program below, and below the results I will post my source code

```
max depth: 0
node: 43 depth: 0
max depth: 1
node: 43 depth: 0
node: 51 depth: 1
node: 44 depth: 1
node: 42 depth: 1
node: 35 depth: 1
max depth: 2
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 45 depth: 2
node: 36 depth: 2
node: 42 depth: 1
node: 41 depth: 2
node: 34 depth: 2
node: 35 depth: 1
max depth: 3
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 58 depth: 3
node: 52 depth: 2
node: 53 depth: 3
node: 44 depth: 3
node: 50 depth: 2
node: 49 depth: 3
node: 42 depth: 3
node: 44 depth: 1
node: 42 depth: 1
node: 35 depth: 1
max depth: 4
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
```

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 61 depth: 4
node: 52 depth: 4
node: 58 depth: 3
node: 57 depth: 4
node: 50 depth: 4
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 45 depth: 2
node: 37 depth: 3
node: 38 depth: 4
node: 36 depth: 4
node: 36 depth: 2
node: 42 depth: 1
node: 41 depth: 2
node: 40 depth: 3
node: 48 depth: 4
node: 32 depth: 4
node: 34 depth: 2
node: 35 depth: 1
max depth: 5
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 53 depth: 5
node: 52 depth: 4
node: 44 depth: 5
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 49 depth: 5
node: 50 depth: 4
node: 42 depth: 5
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 42 depth: 1
node: 35 depth: 1
max depth: 6
node: 43 depth: 0

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 54 depth: 6
node: 53 depth: 5
node: 52 depth: 6
node: 45 depth: 6
node: 52 depth: 4
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 5
node: 50 depth: 6
node: 41 depth: 6
node: 50 depth: 4
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 36 depth: 2
node: 35 depth: 3
node: 34 depth: 4
node: 42 depth: 1
node: 35 depth: 1
max depth: 7
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 55 depth: 7
node: 54 depth: 6
node: 53 depth: 7
node: 53 depth: 5
node: 52 depth: 4
node: 44 depth: 5
node: 36 depth: 6
node: 37 depth: 7

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 35 depth: 7
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 7
node: 40 depth: 7
node: 49 depth: 5
node: 50 depth: 4
node: 42 depth: 5
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 42 depth: 1
node: 35 depth: 1
max depth: 8
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 55 depth: 7
node: 54 depth: 8
node: 47 depth: 8
node: 54 depth: 6
node: 53 depth: 5
node: 52 depth: 6
node: 44 depth: 7
node: 45 depth: 8
node: 36 depth: 8
node: 45 depth: 6
node: 52 depth: 4
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 7
node: 50 depth: 8
node: 41 depth: 8
node: 40 depth: 7
node: 32 depth: 8

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 49 depth: 5
node: 50 depth: 4
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 42 depth: 1
node: 34 depth: 2
node: 35 depth: 1
max depth: 9
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 55 depth: 7
node: 54 depth: 8
node: 53 depth: 9
node: 47 depth: 8
node: 39 depth: 9
node: 54 depth: 6
node: 53 depth: 5
node: 52 depth: 4
node: 44 depth: 5
node: 36 depth: 6
node: 37 depth: 7
node: 35 depth: 7
node: 34 depth: 8
node: 42 depth: 9
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 7
node: 40 depth: 7
node: 32 depth: 8
node: 24 depth: 9
node: 49 depth: 5
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 42 depth: 1

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 35 depth: 1
max depth: 10
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 55 depth: 7
node: 54 depth: 8
node: 53 depth: 9
node: 52 depth: 10
node: 45 depth: 10
node: 47 depth: 8
node: 39 depth: 9
node: 38 depth: 10
node: 31 depth: 10
node: 54 depth: 6
node: 53 depth: 5
node: 52 depth: 4
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 7
node: 50 depth: 8
node: 42 depth: 9
node: 41 depth: 10
node: 34 depth: 10
node: 41 depth: 8
node: 40 depth: 7
node: 32 depth: 8
node: 24 depth: 9
node: 25 depth: 10
node: 16 depth: 10
node: 49 depth: 5
node: 50 depth: 4
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 36 depth: 2
node: 42 depth: 1

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 35 depth: 1
max depth: 11
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 55 depth: 7
node: 54 depth: 8
node: 53 depth: 9
node: 52 depth: 10
node: 44 depth: 11
node: 45 depth: 10
node: 37 depth: 11
node: 47 depth: 8
node: 39 depth: 9
node: 31 depth: 10
node: 23 depth: 11
node: 54 depth: 6
node: 53 depth: 5
node: 52 depth: 4
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 7
node: 50 depth: 8
node: 42 depth: 9
node: 41 depth: 10
node: 40 depth: 11
node: 34 depth: 10
node: 35 depth: 11
node: 41 depth: 8
node: 40 depth: 7
node: 49 depth: 5
node: 50 depth: 4
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 42 depth: 1
node: 35 depth: 1

CPSC 3750
Assignment 1
Luke Leontowich
001217324

max depth: 12
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 55 depth: 7
node: 54 depth: 8
node: 53 depth: 9
node: 52 depth: 10
node: 44 depth: 11
node: 45 depth: 12
node: 36 depth: 12
node: 45 depth: 10
node: 47 depth: 8
node: 39 depth: 9
node: 38 depth: 10
node: 31 depth: 10
node: 23 depth: 11
node: 22 depth: 12
node: 15 depth: 12
node: 54 depth: 6
node: 53 depth: 5
node: 52 depth: 4
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 7
node: 50 depth: 8
node: 42 depth: 9
node: 41 depth: 10
node: 40 depth: 11
node: 32 depth: 12
node: 34 depth: 10
node: 41 depth: 8
node: 40 depth: 7
node: 49 depth: 5
node: 50 depth: 4
node: 52 depth: 2
node: 50 depth: 2

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 44 depth: 1
node: 42 depth: 1
node: 35 depth: 1
max depth: 13
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 55 depth: 7
node: 54 depth: 8
node: 53 depth: 9
node: 52 depth: 10
node: 44 depth: 11
node: 45 depth: 12
node: 37 depth: 13
node: 36 depth: 12
node: 35 depth: 13
node: 45 depth: 10
node: 47 depth: 8
node: 39 depth: 9
node: 31 depth: 10
node: 23 depth: 11
node: 22 depth: 12
node: 21 depth: 13
node: 14 depth: 13
node: 15 depth: 12
node: 7 depth: 13
node: 54 depth: 6
node: 53 depth: 5
node: 52 depth: 4
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 7
node: 50 depth: 8
node: 42 depth: 9
node: 41 depth: 10
node: 40 depth: 11
node: 32 depth: 12

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 24 depth: 13
node: 41 depth: 8
node: 40 depth: 7
node: 49 depth: 5
node: 50 depth: 4
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 42 depth: 1
node: 35 depth: 1
max depth: 14
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 55 depth: 7
node: 54 depth: 8
node: 53 depth: 9
node: 52 depth: 10
node: 44 depth: 11
node: 45 depth: 12
node: 37 depth: 13
node: 38 depth: 14
node: 36 depth: 14
node: 36 depth: 12
node: 45 depth: 10
node: 47 depth: 8
node: 54 depth: 6
node: 53 depth: 5
node: 52 depth: 4
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 7
node: 50 depth: 8
node: 42 depth: 9
node: 41 depth: 10
node: 40 depth: 11
node: 32 depth: 12

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 24 depth: 13
node: 25 depth: 14
node: 16 depth: 14
node: 34 depth: 10
node: 41 depth: 8
node: 40 depth: 7
node: 49 depth: 5
node: 50 depth: 4
node: 52 depth: 2
node: 50 depth: 2
node: 44 depth: 1
node: 42 depth: 1
node: 35 depth: 1
max depth: 15
node: 43 depth: 0
node: 51 depth: 1
node: 59 depth: 2
node: 60 depth: 3
node: 61 depth: 4
node: 62 depth: 5
node: 63 depth: 6
node: 55 depth: 7
node: 54 depth: 8
node: 53 depth: 9
node: 52 depth: 10
node: 44 depth: 11
node: 45 depth: 12
node: 37 depth: 13
node: 38 depth: 14
node: 39 depth: 15
node: 36 depth: 14
node: 35 depth: 15
node: 36 depth: 12
node: 45 depth: 10
node: 47 depth: 8
node: 54 depth: 6
node: 53 depth: 5
node: 52 depth: 4
node: 58 depth: 3
node: 57 depth: 4
node: 56 depth: 5
node: 48 depth: 6
node: 49 depth: 7

CPSC 3750
Assignment 1
Luke Leontowich
001217324

node: 50 depth: 8
node: 42 depth: 9
node: 41 depth: 10
node: 40 depth: 11
node: 32 depth: 12
node: 24 depth: 13
node: 25 depth: 14
node: 17 depth: 15
node: 18 depth: 16

CPSC 3750
Assignment 1
Luke Leontowich
001217324

```
#include <iostream>
#include <utility>
#include <stack>
using namespace std;


//  (i, j) location on grid
typedef pair<int, int> pr;

//  Convert a number to it's (i, j) location
pr getLoc(const int& n) {
    pr p;
    p.first = n / 8 + 1;
    p.second = n % 8 + 1;
    return p;
}

//  convert (i, j) location to node number
int getNum(const pr& p) {
    return (p.first-1) * 8 + (p.second -1);
}

//  Node
typedef struct {
    int num;
    int depth;
} Node;


int main () {
    //  Number the nodes
    int arr[10][10];
    int k = 0;
    for (int i = 1; i < 9; ++i) {
        for (int j = 1; j < 9; ++j) {
            arr[i][j] = k++;
        }
    }


    //  goal state (node.num == 18)
    bool goal = false;
```

CPSC 3750
Assignment 1
Luke Leontowich
001217324

```
//  iterative deepening depth-first search
for (int depth = 0; depth < 20; ++depth) {
   //  if goal state found then we are done
   if (goal) {
      break;
   }

   //  print the current max depth
   cout << "max depth: " << depth << "\n";


   stack<Node> s;

   //  initialize start node
   Node node;
   node.num = 43;
   node.depth = 0;
   s.push(node);

   //  initialize the checked state
   bool checked[10][10];
   for (int i = 0; i < 10; ++i) {
      for (int j = 0; j < 10; ++j) {
         //  border around the board
         if (i == 0 || j == 0 || i == 9 || j == 9) {
            checked[i][j] = true;
         } else {
            checked[i][j] = false;
         }
      }
   }

   //  set the marked out position and start node to already checked
   pr p = getLoc(43);
   checked[p.first][p.second] = true;
   p = getLoc(26);
   checked[p.first][p.second] = true;
   p = getLoc(27);
   checked[p.first][p.second] = true;
   p = getLoc(28);
   checked[p.first][p.second] = true;
   p = getLoc(29);
   checked[p.first][p.second] = true;
```

CPSC 3750
Assignment 1
Luke Leontowich
001217324

```
p = getLoc(30);
checked[p.first][p.second] = true;
p = getLoc(33);
checked[p.first][p.second] = true;
p = getLoc(46);
checked[p.first][p.second] = true;

//  conduct search
while (!s.empty()) {
    auto current = s.top();
    s.pop();

    //  don't print the nodes that we can't explore
    if (current.depth <= depth) {
        cout << "node: " << current.num << " depth: " << current.depth << "\n";
    }
    //  get (i,j) location of current node
    p = getLoc(current.num);

    //  set current node to explored to prevent cycles

    //  mark the current node to checked
    checked[p.first][p.second] = true;

    //  if goal state then we are done
    if (current.num == 18) {
        goal = true;
         cout << "node: " << current.num << " depth: " << current.depth << "\n";
        break;

    //  if we have not met our depth limit
    } else if (current.depth <= depth) {
        p = getLoc(current.num);
        //  add up
        if (!checked[p.first-1][p.second]) {
            pr temp;
            temp.first = p.first-1;
            temp.second = p.second;
            node.num = getNum(temp);
            node.depth = current.depth+1;
            s.push(node);
        }
        // add left
```

CPSC 3750
Assignment 1
Luke Leontowich
001217324

```
            if (!checked[p.first][p.second-1]) {
               pr temp;
               temp.first = p.first;
               temp.second = p.second-1;
               node.num = getNum(temp);
               node.depth = current.depth+1;
               s.push(node);
            }
            //  add right
            if (!checked[p.first][p.second+1]) {
               pr temp;
               temp.first = p.first;
               temp.second = p.second +1;
               node.num = getNum(temp);
               node.depth = current.depth+1;
               s.push(node);
            }
            //  add down
            if (!checked[p.first+1][p.second]) {
               pr temp;
               temp.first = p.first+1;
               temp.second = p.second;
               node.num = getNum(temp);
               node.depth = current.depth+1;
               s.push(node);
            }

         //  case that node is past the depth limit, then do nothing
         } else {
            ;
         }
      }
   }

   return 0;
}
```